# USB and PS/2 Multimedia Keyboard Interface

Designer Reference Manual

**M68HC08**
**Microcontrollers**

DRM014
Rev. 1
4/2006

freescale.com

freescale™
semiconductor

# USB and PS/2 Multimedia Keyboard Interface

**Designer Reference Manual**

by: Derek Lau
Freescale Semiconductor, Inc.
Hong Kong

To provide the most up-to-date information, the revision of our documents on the World Wide Web will be the most current. Your printed copy may be an earlier revision. To verify that you have the latest information available, refer to http://www.freescale.com

The following revision history table summarizes changes contained in this document. For your convenience, the page number designators have been linked to the appropriate location.

## Revision History

| Date | Revision Level | Description | Page Number(s) |
|---|---|---|---|
| 04/2006 | 1 | Replaced unclear schematic. | 26 |
| 03/2003 | 0 | Initial release | N/A |

# Table of Contents

## Chapter 1
## USB and PS/2 Multimedia Keyboard Interface


1.1 Introduction . . . . . 7
1.2 Overview . . . . . 7
1.3 44-pin QFP MC68HC908JB8 Features . . . . . 7
1.4 Hardware Descriptions . . . . . 8
1.4.1 Key Matrix . . . . . 9
1.4.2 In-Circuit Programming . . . . . 10
1.5 Firmware Description . . . . . 11
1.5.1 USB and PS/2 Detection . . . . . 12
1.5.2 PS/2 Main Routine . . . . . 12
1.5.3 PS/2 Protocol . . . . . 12
1.5.4 Host to Device Communications . . . . . 14
1.5.5 Device to Host Communications . . . . . 14
1.5.6 PS/2 Keyboard Command . . . . . 15
1.5.7 Host to Keyboard Commands . . . . . 15
1.5.8 Keyboard to Host Commands . . . . . 16
1.5.9 PS/2 Scan Codes . . . . . 16
1.5.10 USB Main Routine . . . . . 17
1.6 Firmware Files . . . . . 24
1.7 Test Description . . . . . 24
1.8 Customization . . . . . 24
1.8.1 Hardware . . . . . 24
1.8.2 Firmware . . . . . 25
1.9 Extra Features . . . . . 25
1.10 Further Information . . . . . 25
1.10.1 Related Documents . . . . . 25
1.11 Schematics . . . . . 25


## Appendix A.
## Glossary

# Chapter 1
# USB and PS/2 Multimedia Keyboard Interface

## 1.1 Introduction

This manual describes a reference design of a Universal Serial Bus and PS/2 multimedia keyboard interface for Microsoft Windows by using the MC68HC908JB8.

For the full MC68HC908JB8 specification, please refer to the data sheet, Freescale order number: MC68HC908JB8/D.

## 1.2 Overview

The Freescale MC68HC908JB8 is a member of the HC08 Family of microcontrollers (MCUs). The features of the MC68HC908JB8 include a configured Universal Serial Bus (USB) or PS/2 interface, which makes this MCU suited for personal computer Human Interface Devices (HID), such as mice and keyboards. The MC68HC908JB8 is available in several packages to fit into various applications. A multimedia keyboard with USB and PS/2 interface is demonstrated using the MC68HC908JB8 with a 44-pin QFP package. The main features of the keyboard include:

- Fully USB specification 1.1 compliant
- USB or PS/2 interface auto detect
- Windows 98, ME and 2000 compatible
- Power management keys (power, wake and sleep) support
- Multimedia key support
- In-circuit programming for firmware modification

## 1.3 44-pin QFP MC68HC908JB8 Features

The 44-pin QFP MC68HC908JB8 is targeted for USB and PS/2 interface keyboard applications with minimum external components needed. Features include:

- USB D+ and D– pins shared with PS/2 data and clock pins
- 27 out of 37 I/O pins with internal pull-up supports up to 8x19 key matrix
- 10mA direct drive pins for the Num Lock, Caps Lock and Scroll Lock LEDs
- Internal 1.5K pull-up for USB D– data line
- Internal 5K pull-ups for PS/2 data and clock pins

## 1.4 Hardware Descriptions

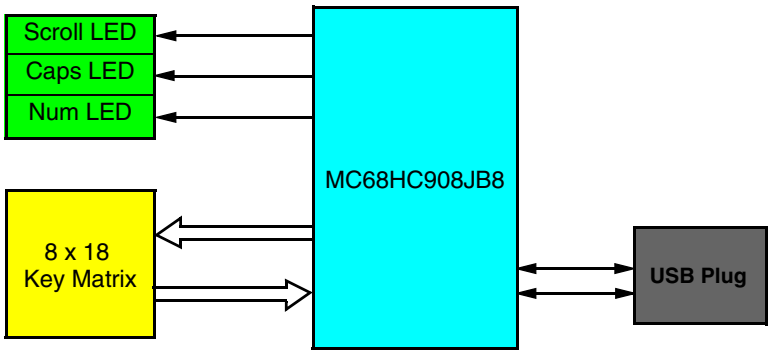


**Figure 1-1. . Block Diagram**

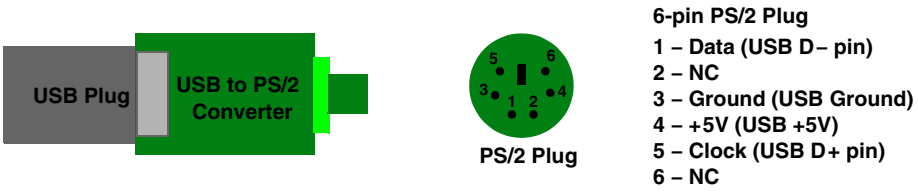


**Figure 1-2. USB and PS/2 Connections**

Figure 1-1 shows the block diagram of the keyboard. The solution includes the JB8, key button inputs and LED indicator outputs only. The connections of the corresponding USB and PS/2 signals are shown in Figure 1-2 The USB to PS/2 converter standard connections are the USB D– and D+ pins connected to the PS/2 Data and Clock pins respectively.

Figure 1-3. shows the printed circuit board and Figure 1-16 shows the schematic of the keyboard.

- J1 is used for USB connection
- J2 is used for PS/2 mouse connection (for future development)
- J3 and J7 are used for in-circuit programming
- 8 rows x 18 columns key matrix is implemented
  - 8 rows implemented in PTA[7:0]
  - 18 columns implemented in PTB[7:0], PTC[7:0], PTE0 and PTE2
- Keyboard LEDs
  - Scroll Lock at PTD2
  - Caps Lock at PTD3
  - Num Lock at PTD4

Figure 1-4 shows the key matrix for the 107 standard keyboard with power management keys (Power, Wake and Sleep). Figure 1-5 shows the key matrix for multimedia keyboard with function key.

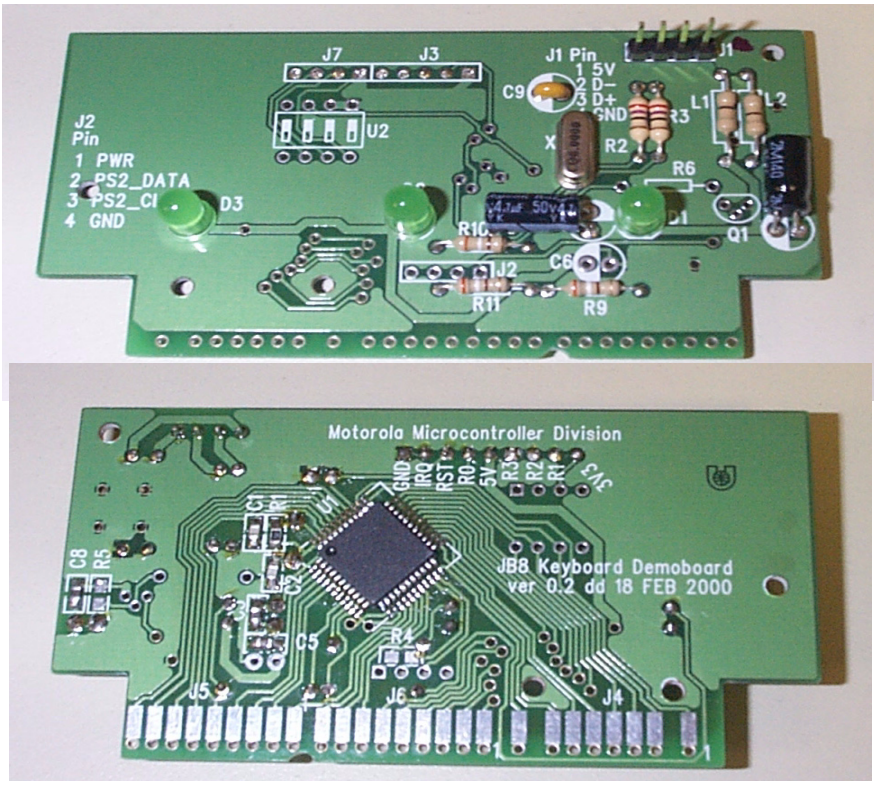


**Figure 1-3. Keyboard PCB**

## 1.4.1 Key Matrix

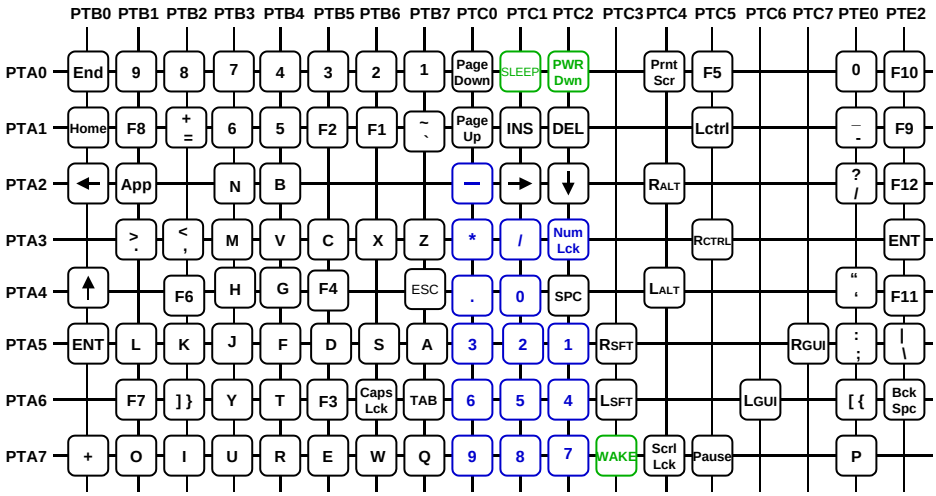
PTB0 PTB1 PTB2 PTB3 PTB4 PTB5 PTB6 PTB7 PTC0 PTC1 PTC2 PTC3 PTC4 PTC5 PTC6 PTC7 PTE0 PTE2
PTA0 End 9 8 7 4 3 2 1 Page Down SLEEP PWR Dwn Prnt Scr F5 0 F10
PTA1 Home F8 + = 6 5 F2 F1 ~ ` Page Up INS DEL Lctrl _ - F9
PTA2 App N B RALT ? / F12
PTA3 > . < , M V C X Z * / Num Lck RCTRL ENT
PTA4 F6 H G F4 ESC . 0 SPC LALT " ' F11
PTA5 ENT L K J F D S A 3 2 1 RSFT RGUI : ; | \
PTA6 F7 ] } Y T F3 Caps Lck TAB 6 5 4 LSFT LGUI [ { Bck Spc
PTA7 + O I U R E W Q 9 8 7 WAKE Scrl Lck Pause P

**Figure 1-4. 107-Key Matrix**

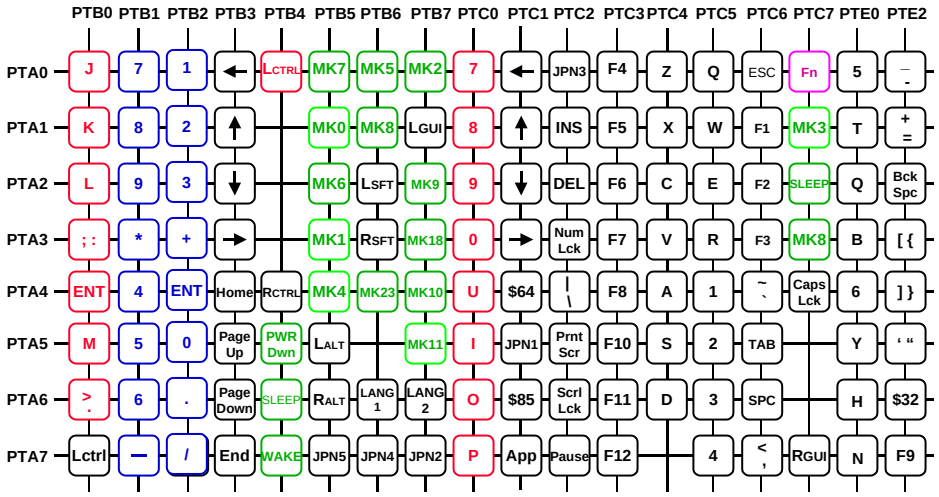
PTB0 PTB1 PTB2 PTB3 PTB4 PTB5 PTB6 PTB7 PTC0 PTC1 PTC2 PTC3 PTC4 PTC5 PTC6 PTC7 PTE0 PTE2
PTA0 J 7 1 ← LCTRL MK7 MK5 MK2 7 ← JPN3 F4 Z Q ESC Fn 5 _ -
PTA1 K 8 2 ↑ MK0 MK8 LGUI 8 ↑ INS F5 X W F1 MK3 T + =
PTA2 L 9 3 ↓ MK6 LSFT MK9 9 ↓ DEL F6 C E F2 SLEEP Q Bck Spc
PTA3 ; : * + → MK1 RSFT MK18 0 → Num Lck F7 V R F3 MK8 B [ {
PTA4 ENT 4 ENT Home RCTRL MK4 MK23 MK10 U $64 | \ F8 A 1 ~ ` Caps Lck 6 ] }
PTA5 M 5 0 Page Up PWR Dwn LALT MK11 I JPN1 Prnt Scr F10 S 2 TAB Y ' "
PTA6 > . 6 . Page Down SLEEP RALT LANG 1 LANG 2 O $85 Scrl Lck F11 D 3 SPC H $32
PTA7 Lctrl — / End WAKE JPN5 JPN4 JPN2 P App Pause F12 4 < , RGUI N F9

**Figure 1-5. Multimedia Key Matrix**

## 1.4.2 In-Circuit Programming

J3 and J7 contain all the signals for the MC68HC908JB8 to enter monitor mode for In-Circuit Programming. The ICP requires J3 and J7 to be connected with a cable to the ICP adaptor board, which is plugged-into the M68HC08 Serial Programmer (M68SPGRM08) (see Figure 1-6). Running the MCUScribe software on the PC allows erase, programming and verification of the firmware in the MC68HC908JB8. The communication baud rate is determined by the jumper setting of J3 of the ICP adaptor board. Connect pins 1 and 2 of J3 for 9600bps and pins 2 and 3 for 19200bps.

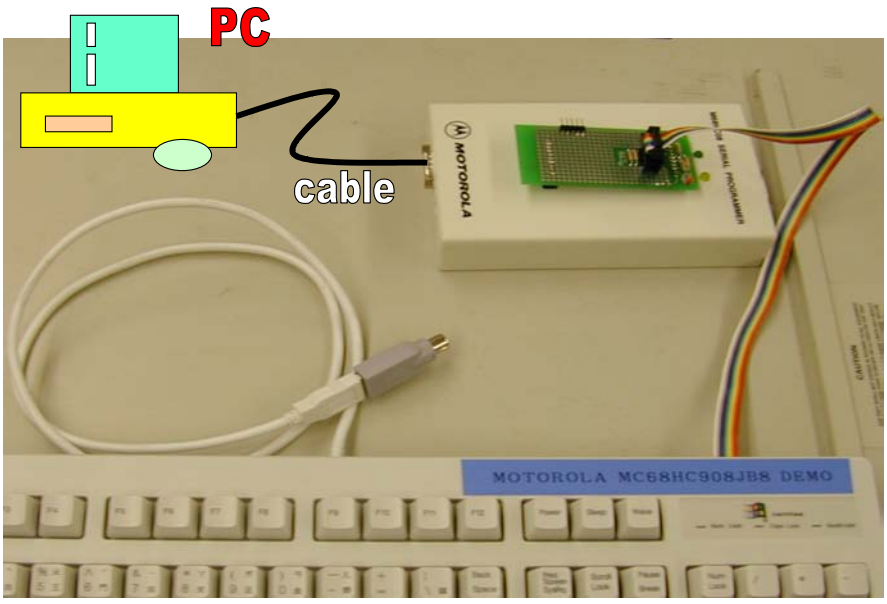


**Figure 1-6. In-Circuit Programming Connection**

## 1.5 Firmware Description

The firmware consists of three main parts:

- USB and PS/2 interface detection
- PS/2 main program and subroutines
- USB main program and subroutines

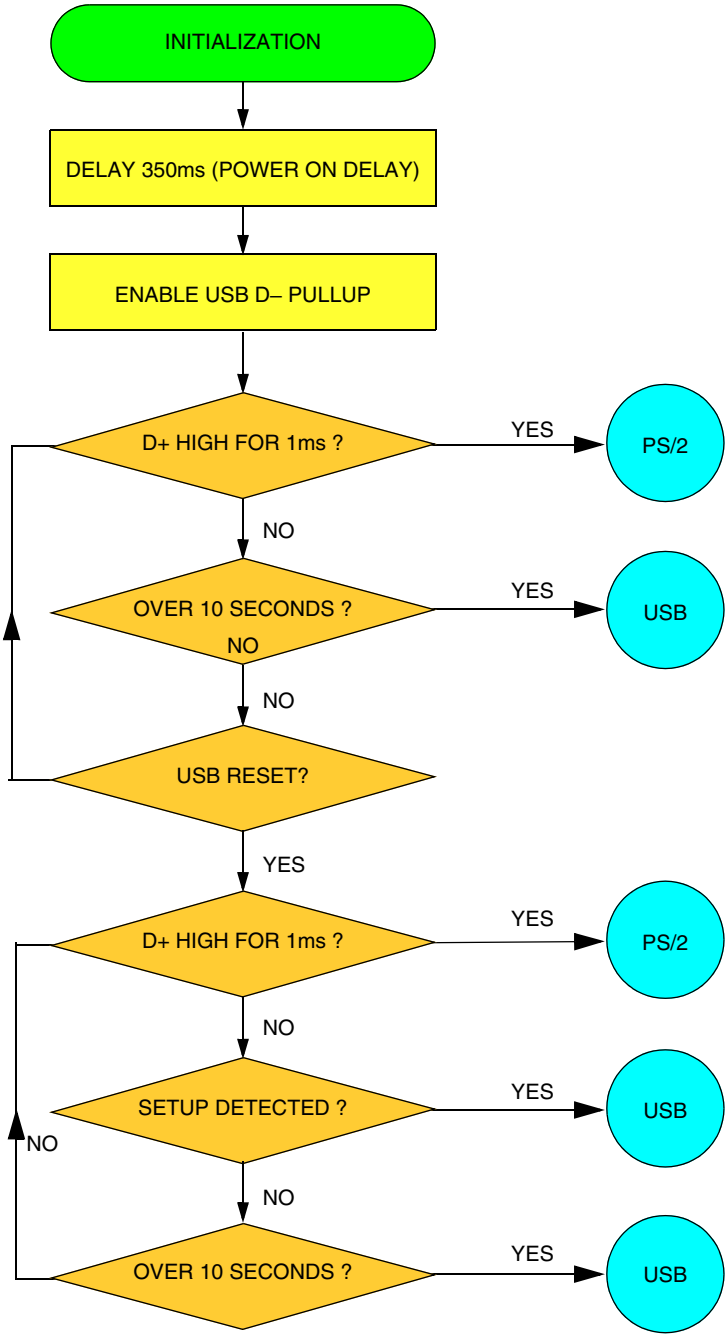


**Figure 1-7. USB and PS/2 Detection**

### 1.5.1 USB and PS/2 Detection

PTE3 and PTE4 can be configured as USB D+ and D– pins or as open-drain I/O pins for PS/2 data and clock lines. Figure 1-7 shows the algorithm for distinguishing between a USB and a PS/2 interface.

After power on, the interface protocol is undetermined and can be either a USB or a PS/2 interface. The firmware first initializes the registers and the I/O ports. It then performs a 350ms software delay to meet the PS/2 power on delay requirement. The USB engine and the internal USB pullup resistor are enabled. PTE3 and PTE4 are configured as USB D+ and D– pins with a 1.5K internal pullup at D– pin.

For a low speed USB interface, D+ pin will not be continuously high for more than 4μs because of the bit stuffing mechanism. The PS/2 interface is determined by 1ms continuously high at D+ pin while the USB interface is determined by receiving a SETUP token.

After the interface mode is detected. the firmware enters either the PS/2 or USB main routine.

### 1.5.2 PS/2 Main Routine

In PS/2 mode, PTE3 and PTE4 are configured as open-drain I/O pins with 5K internal pullup resistors enabled. Figure 1-8 shows the PS/2 main routine. The main functions of the PS/2 routines are:

- Receive commands from host
- Respond to received commands
- Scan key matrix
- Send make code to host if key pressed
- Send break code to host if key released

### 1.5.3 PS/2 Protocol

The PS/2 is a bidirectional serial interface using two signals: Clock and Data. The data consists of 11 bits including 1 start bit, 8 data bits, 1 odd parity bit, and 1 stop bit. PS/2 device generates the clock signal with a typical cycle of 80μs in both host-to-device or device-to-host communications.

**Table 1-1. Clock and Data Line Control**

| | Register Bits | | | | | | | Port Pins | |
|---|---|---|---|---|---|---|---|---|---|
| **Register** | **POCR** | **PTE** | **DDRE** | **IOCR** | **POCR** | **PTE** | **DDRE** | **PTE4 (Data)** | **PTE3 (Clock)** |
| **Bit** | **PTE4P** | **PTE4** | **DDRE4** | **PTE4IE** | **PTE3P** | **PTE3** | **DDRE3** | **PTE4 (Data)** | **PTE3 (Clock)** |
| **PTE4 Interrupt Enabled** | — | — | — | 1 | — | — | — | Input High | — |
| **Release Data/Clock High** | 1 | — | 0 | — | 1 | — | 0 | Input High | Input High |
| **Drive Data/Clock Low** | — | 0 | 1 | — | — | 0 | 1 | Output Low | Output Low |

Table 1-1 shows the setting of the registers for controlling the PTE3 (Clock) and the PTE4 (Data) pins. Instead of setting DDR3/DDR4 as high to output a high signal, we set DDR3/DDR4 as an input with internal pullup to perform the same function. In the PS/2 routines, the values of PTE3 and PTE4 are cleared to zero and the values of PTE3P and PTE4P are set to one. Set DDR3 or DDR4 to one to force it as output low, or clear DDR3 or DDR4 to zero to make it high impedance and pulled high by the 5K internal resistor.

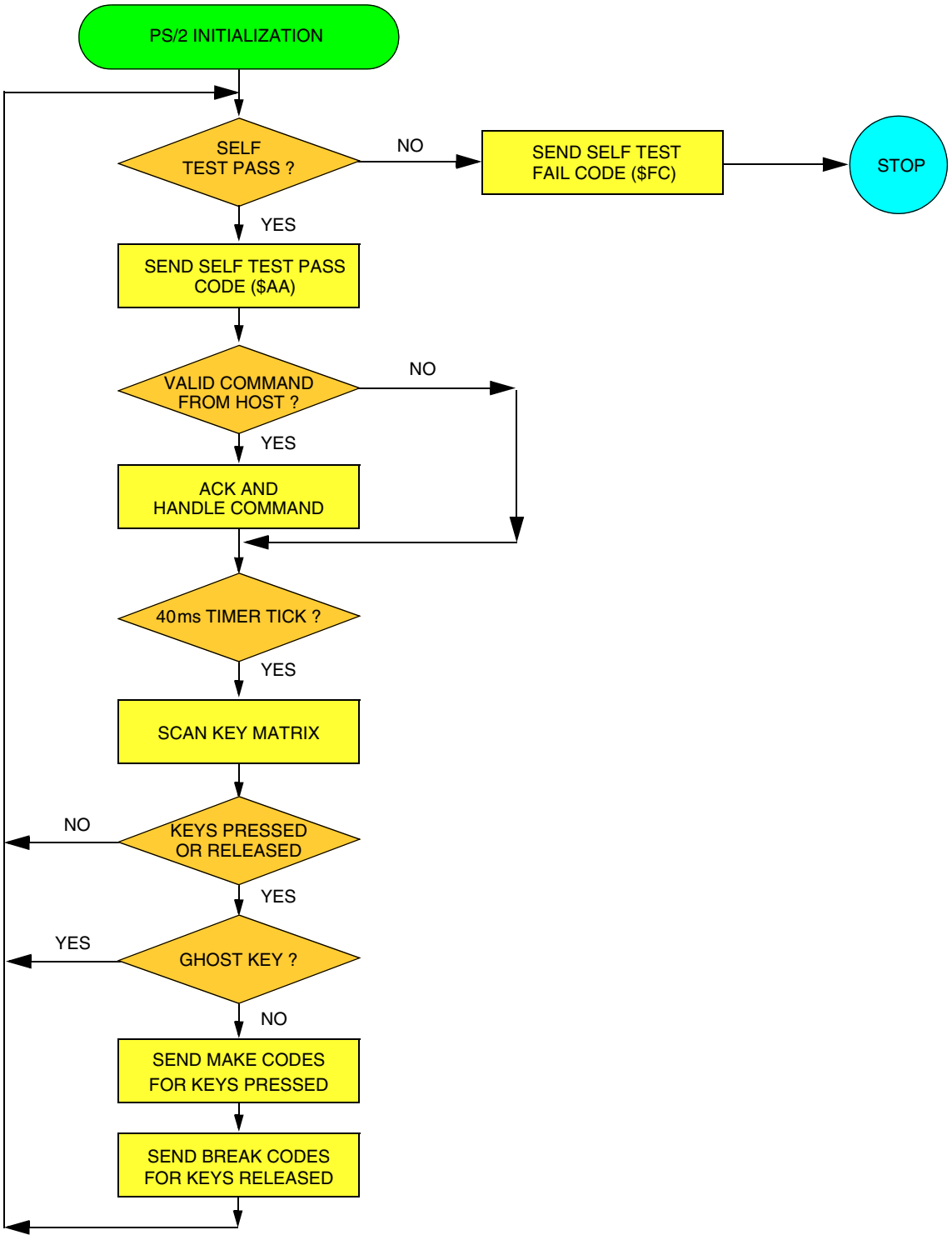


**Figure 1-8. PS/2 Main Routine**

### 1.5.4 Host to Device Communications

Data sent from host to device is read while the clock line is high. In an idle state, both the Clock and the Data lines are pulled high. The host starts sending data by pulling the Clock line low for a minimum of 100ms. Figure 1-9 shows the signal diagram. Communications steps are shown as below:

1. Host waits until no auxiliary device transmission is in progress.
2. Host pulls the Clock line low.
3. Host pulls the Data line low as the start bit.
4. Host releases the Clock line.
5. Device pulls Clock line low.
6. Host sends out data.
7. Device releases the Clock line high and read Data.
8. Device reads the Clock line and aborts communication if the Clock line is low.
9. Repeat steps 5 to 7 for Data 1 to Data 7 and the parity bit.
10. Device pulls the Clock line low.
11. Host releases the Data line.
12. Device releases the Clock line high.
13. Device reads the Data line high for stop signal and sends error if the Data line is low.
14. Device pull the Data and the Clock lines low.
15. Device release the Clock and the Data lines.

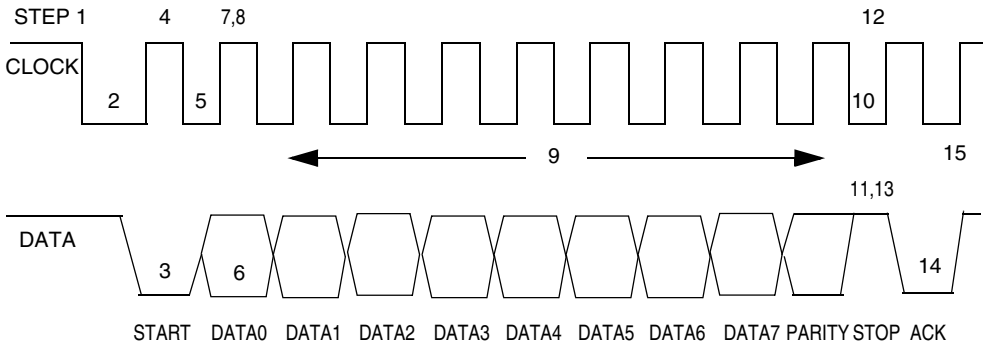


**Figure 1-9. Host to Device Communication**

KBD_IN is the IRQ1 interrupt routine for receiving data from host. The interrupt is configured to execute when a falling edge at the PTE4 (Data) pin is detected.

### 1.5.5 Device to Host Communications

Data sent from device to host is read at the falling edge of the clock. The device checks whether the host is ready by detecting the clock high before transmitting data. Figure 1-10 shows the signal diagram. Communications steps are shown as below:

1. Device waits for the Clock line high for a minimum of 50ms.
2. Abort if the Data line is low.
3. Device sends out data.
4. Device pulls the Clock line low.
5. Device releases the Clock line high.

6. Device reads the Clock line and aborts communication if the Clock line is low.
7. Repeat steps 4 to 6 for Data 0 to Data 7, the parity bit and the stop bit.
8. Device releases the Clock line high.

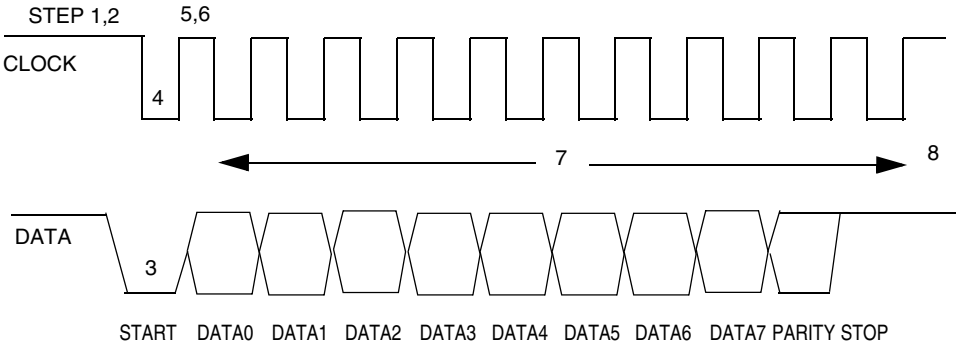


**Figure 1-10. Device to Host Communication**

KBD_OUT is the routine for transmitting data to the host. The data to be transmitted is put into V_TxByte before calling this routine.

### 1.5.6 PS/2 Keyboard Command

Both the host and the keyboard may send commands to each other. The keyboard transmits an acknowledge ($FA) after receiving a valid command. When the keyboard receives an invalid command, it returns a resent command ($FA).

### 1.5.7 Host to Keyboard Commands

**Table 1-2. Host to Keyboard Commands**

| Code | Description | Implemented |
|---|---|---|
| $ED | Set status indicators | YES |
| $EE | Echo | YES |
| $F0 | Set alternate Scan Code | NO |
| $F2 | Get keyboard ID | YES |
| $F3 | Set typematic repeat rate | YES |
| $F4 | Enable Scan | YES |
| $F5 | Disable Scan | YES |
| $F6 | Set default values | YES |
| $F7 | Set all keys typematic | NO |
| $F8 | Set all keys make/break | NO |
| $F9 | Set all keys make | NO |
| $FA | Set key type typematic/make/break | NO |
| $FB | Set key type typematic | NO |
| $FC | Set key type make/break | NO |
| $FD | Set key type make | NO |
| $FE | Resent the last command | YES |
| $FF | Reset | YES |

### 1.5.8 Keyboard to Host Commands

**Table 1-3. Keyboard to Host Commands**

| Code | Description | Implemented |
|---|---|---|
| $00 | Keyboard detection or overrun error | YES |
| $AA | Basic assurance test passed | YES |
| $EE | Echo | YES |
| $FA | Acknowledge | YES |
| $FE | Resend | YES |

### 1.5.9 PS/2 Scan Codes

There are three sets of scan codes (code 1, 2, and 3). Most PCs support scan code set 2, hence this is the only scan code the firmware supports. Make code or break code is sent when any key is pressed or released. While a key is pressed, its make code is sent out repeatedly and the rate depends on the typematic repeat value.

In addition to the IBM standard, Microsoft has published standards for the Windows keys, Power Management keys (sleep, wake, and power), and the Audio Control keys.

**Table 1-4. Scan codes supported by Windows 98, ME, and 2000**

| Description | Make Code | Break Code |
|---|---|---|
| Left Windows | $E0, $1F | $E0, $F0, $1F |
| Right Windows | $E0, $27 | $E0, $F0, $27 |
| Applications | $E0, $2F | $E0, $F0, $2F |
| Power | $E0, $37 | $E0, $F0, $37 |
| Sleep | $E0, $3F | $E0, $F0, $3F |
| Wake | $E0, $5E | $E0, $F0, $5E |

**Table 1-5. Scan codes supported by Windows ME and 2000**

| Description | Make Code | Break Code |
|---|---|---|
| Scan next track | $E0, $4D | $E0, $F0, $4D |
| Scan previous track | $E0, $15 | $E0, $F0, $15 |
| Stop | $E0, $3B | $E0, $F0, $3B |
| Play/Pause | $E0, $34 | $E0, $F0, $34 |
| Mute | $E0, $23 | $E0, $F0, $23 |
| Volume increase | $E0, $32 | $E0, $F0, $32 |
| Volume decrease | $E0, $21 | $E0, $F0, $21 |
| AL Email Reader | $E0, $48 | $E0, $F0, $48 |
| AC search | $E0, $10 | $E0, $F0, $10 |
| AC Home | $E0, $3A | $E0, $F0, $3A |
| AC Forward | $E0, $30 | $E0, $F0, $30 |
| AC Stop | $E0, $28 | $E0, $F0, $28 |
| AC Refresh | $E0, $20 | $E0, $F0, $20 |
| AC Bookmarks | $E0, $18 | $E0, $F0, $18 |

**Table 1-6. Scan codes supported by Windows ME**

| Description | Make Code | Break Code |
|---|---|---|
| AC Calculator | $E0, $2B | $E0, $F0, $2B |
| AC Local Browser | $E0, $40 | $E0, $F0, $40 |
| AC Consumer Control Configuration | $E0, $50 | $E0, $F0, $50 |

### 1.5.10 USB Main Routine

Figure 1-11 shows the USB main routine. The routine scans the keyboard every 40 ms. If there are keys pressed or released, it puts the key codes into a buffer and prepares the input reports for the keys through endpoint 1 or endpoint 2. If the USB bus idles for more than 6 ms, the routine puts the MC68HC908JB8 into STOP mode until it detects a resume signal from the host or any key pressed for remote wake-up.

Figure 1-12 shows the USB interrupt routines. The USB engine automatically responds to a valid USB token with either ACK, NAK, or STALL, depending on the registers setting, and ignores it if it is invalid. The firmware has to set the registers for the USB engine to give a correct response to the token in different stages. The USB interrupt will be executed whenever there is an EOP, resume signal from host, valid data received or data transmitted. The USB interrupt routine also makes preparation for the next USB transaction and handles any valid command or data received.

Figure 1-13 to Figure 1-15 show the routines of handling the Control Transfers. Control transfers have two or three transaction stages: Setup, Data (optional) and Status as shown below:

- Control Write: SETUP, OUT, OUT, OUT... IN
- Control Read: SETUP, IN, IN, IN... OUT
- No Data Control: SETUP, IN

The firmware first distinguishes the kinds of control transfers and does the corresponding preparation for the next stage.

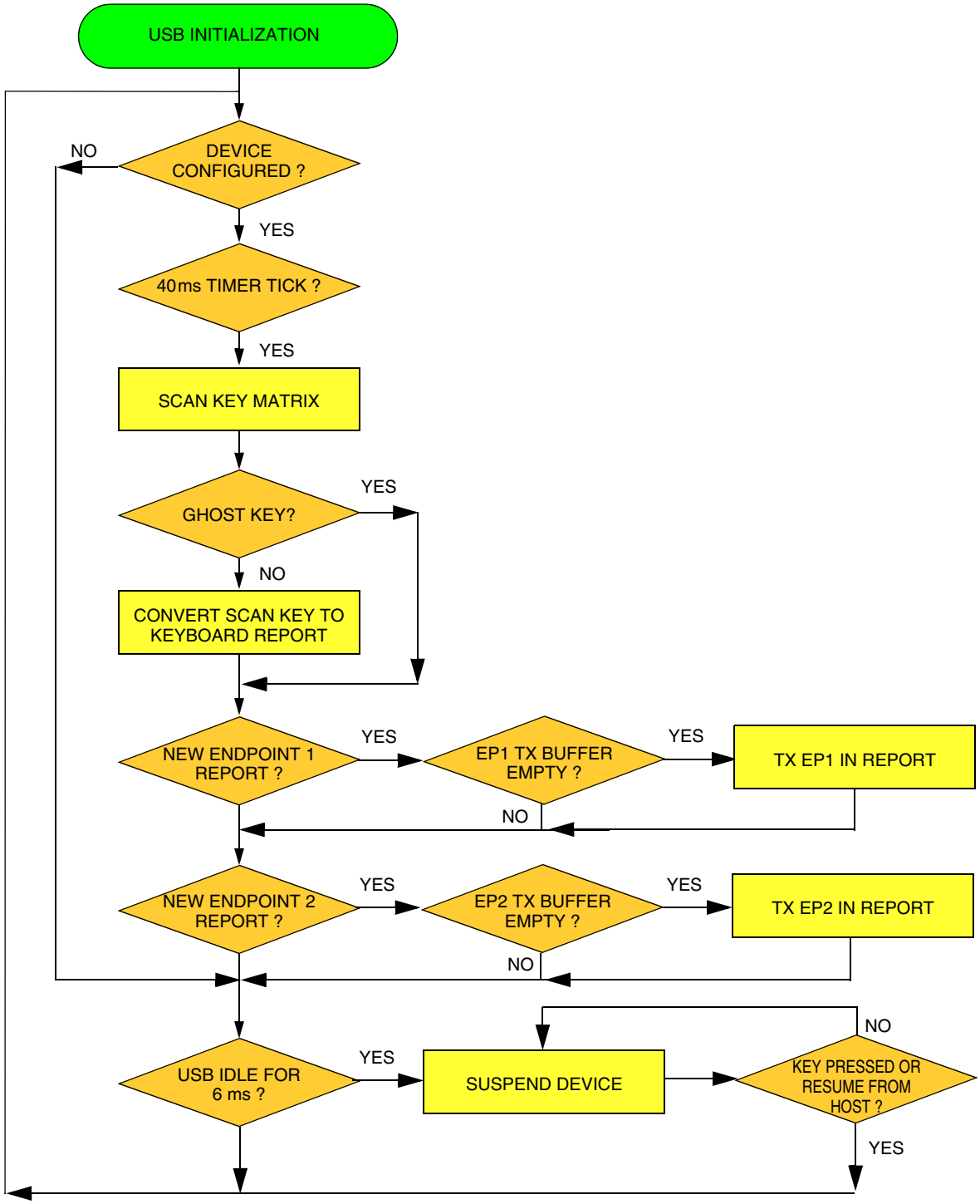


**Figure 1-11. USB Main Routine**

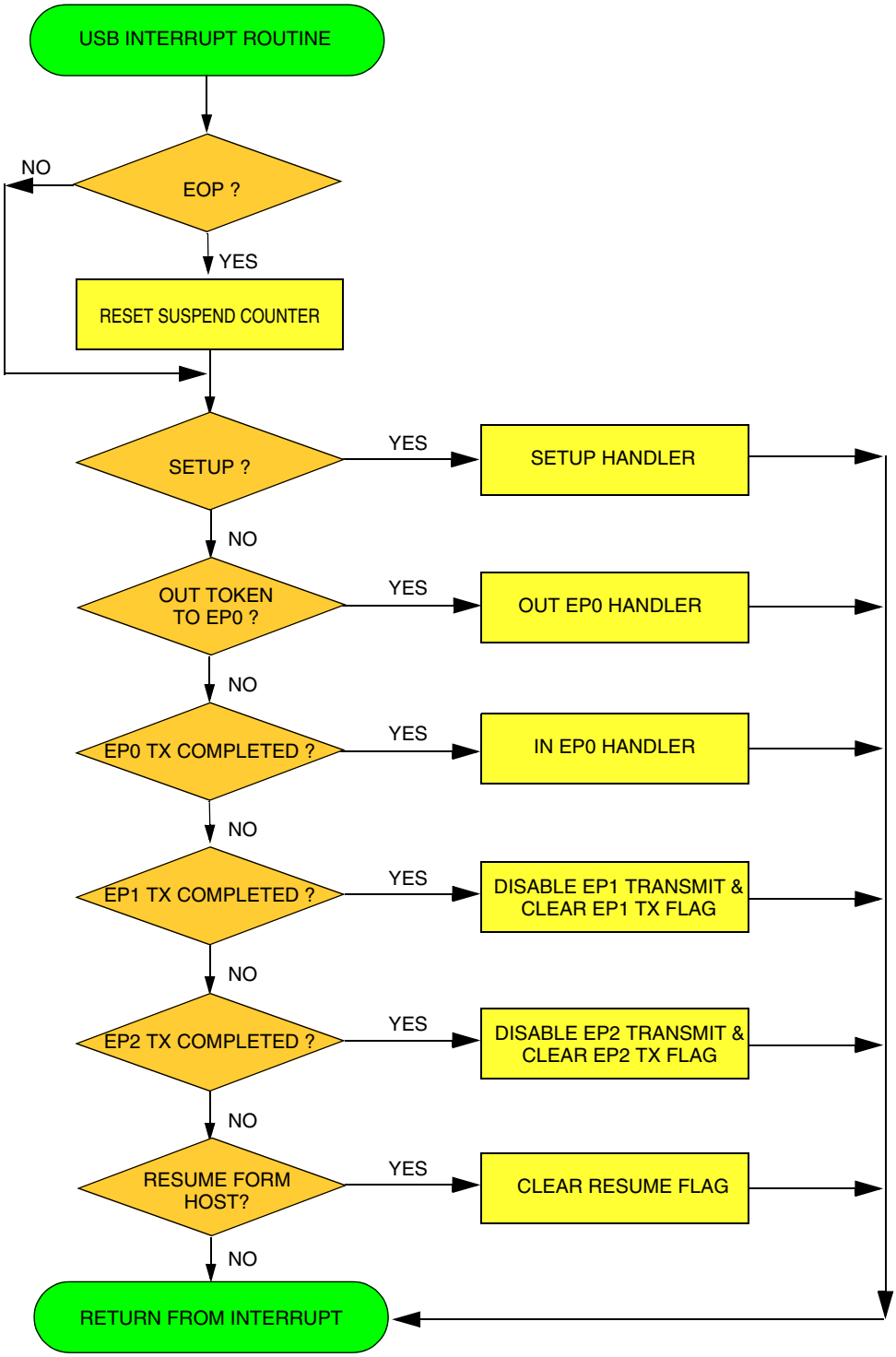


**Figure 1-12. USB Interrupt Routine**

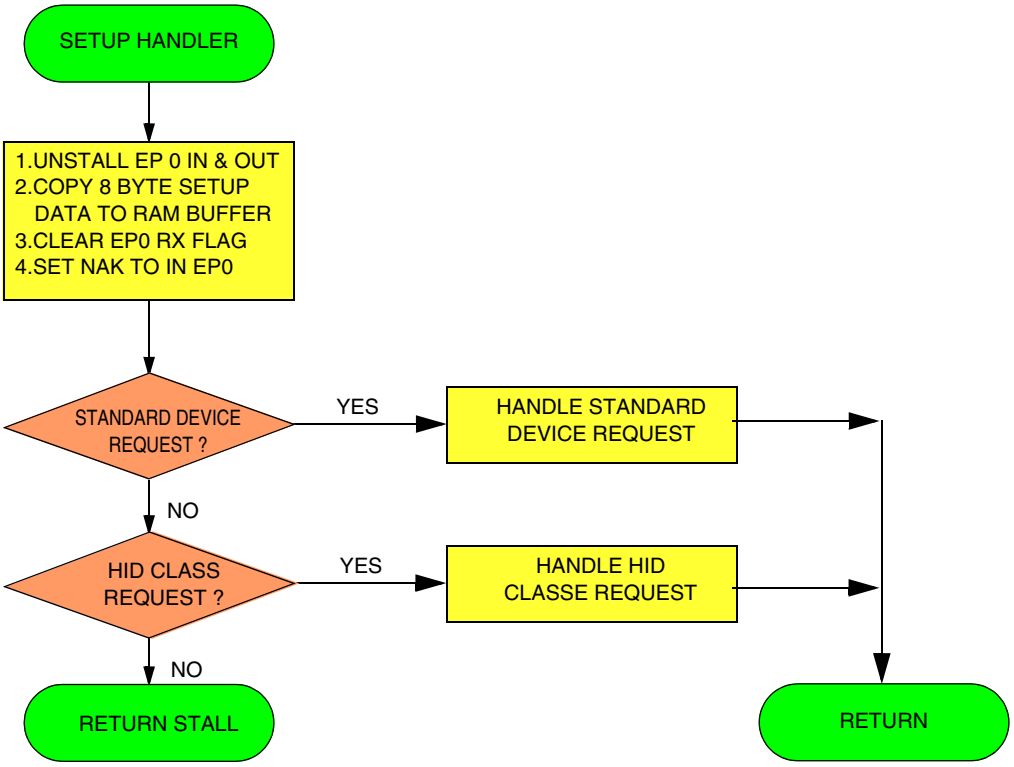


**Figure 1-13. Setup Routine**

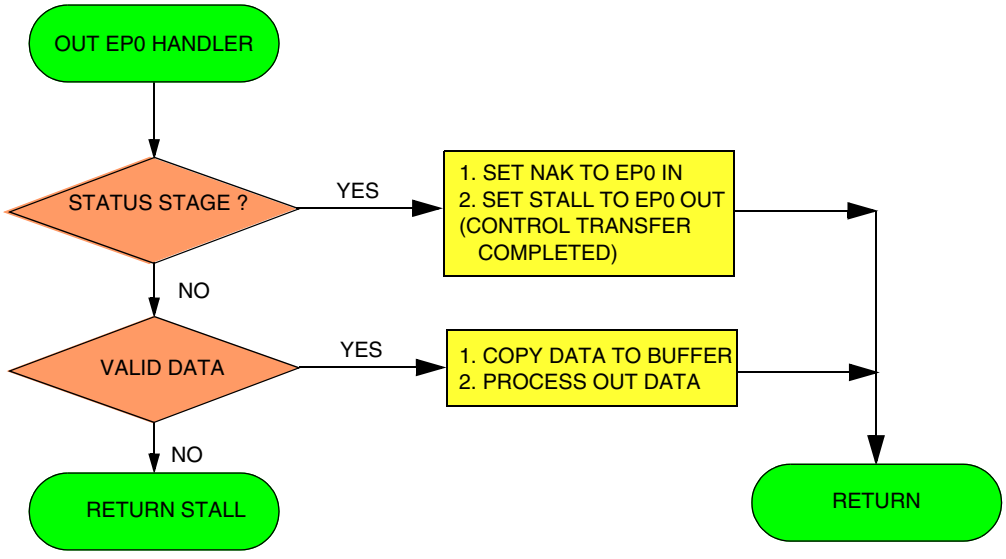


**Figure 1-14. OUT EP0 HANDLER**

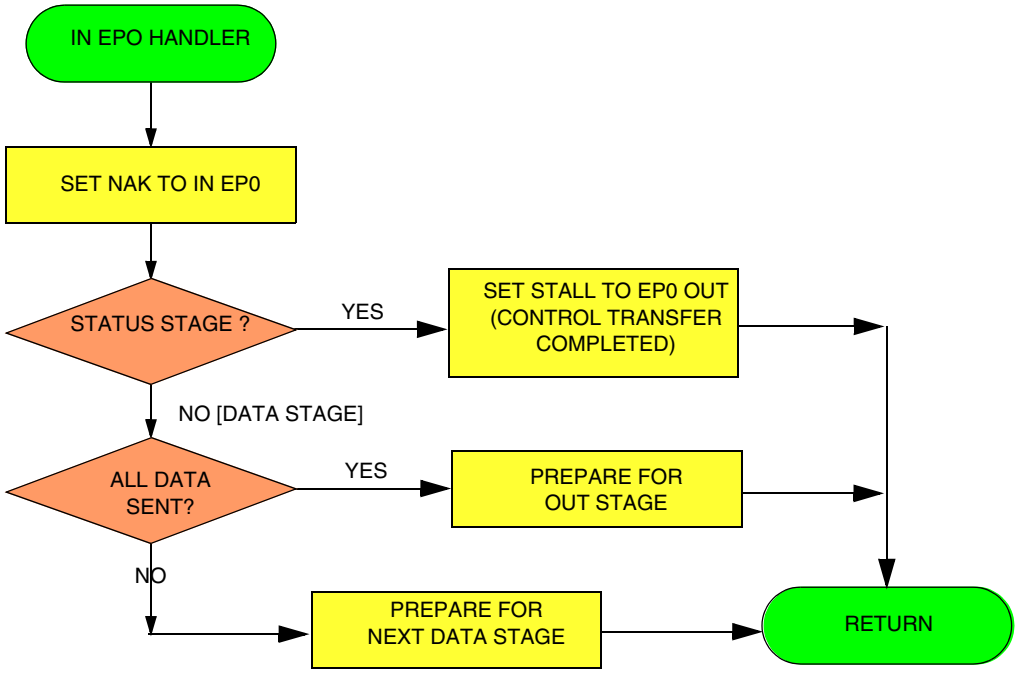


**Figure 1-15. . IN EPO HANDLER**

## USB Key Codes

The key codes or usage IDs for a basic 104 keyboard are defined in the USB HID Usage Tables. In addition to the basic key codes, Microsoft has published standards for the Windows keys, Power Management keys (sleep, wake, and power), and the audio control keys.

## ACPI Power Management Control

**Table 1-7. Consumer Page HID Controls in Windows ME and 2000**

| USAGE | USAGE NAME | DATA TYPE |
|---|---|---|
| $81 | Power | Relative |
| $82 | Sleep | Relative |
| $83 | Wake | Relative |

## Consumer Page Audio Control

**Table 1-8. Consumer Page HID Controls in Windows ME and 2000**

| USAGE | USAGE NAME | USAGE TYPE | DATA TYPE |
|---|---|---|---|
| $B5 | Scan Next Track | One Shot Control | Relative |
| $B6 | Scan Previous Track | One Shot Control | Relative |
| $B7 | Stop | One Shot Control | Relative |
| $CD | Play/Pause | One Shot Control | Relative |
| $E0 | Volume | Linear Control | Relative |
| $E2 | Mute | On/Off Control | Relative |
| $E3 | Bass | Linear Control | Relative |

**Table 1-8. Consumer Page HID Controls in Windows ME and 2000 (Continued)**

| USAGE | USAGE NAME | USAGE TYPE | DATA TYPE |
|---|---|---|---|
| $E | Treble | Linear Control | Relative |
| $E5 | Bass Boost | On/Off Control | Relative |
| $E9 | Volume Increment | Re-Trigger Control | Absolute |
| $EA | Volume Decrement | Re-Trigger Control | Absolute |
| $152 | Bass Increment | Re-Trigger Control | Absolute |
| $153 | Bass Decrement | Re-Trigger Control | Absolute |
| $154 | Treble Increment | Re-Trigger Control | Absolute |
| $155 | Treble Decrement | Re-Trigger Control | Absolute |
| $18A | AL Email Reader | Selector | Relative |
| $221 | Bass Increment | Selector | Relative |
| $223 | Bass Increment | Selector | Relative |
| $224 | Bass Increment | Selector | Relative |
| $225 | Bass Increment | Selector | Relative |
| $226 | Bass Increment | Selector | Relative |
| $227 | Bass Increment | Selector | Relative |
| $183 | AL Consumer Control[1] Configuration | Selector | Relative |
| $192 | AL Calculator[1] | Selector | Relative |
| $194 | AL Local Browser[1] | Selector | Relative |

NOTES:
1. Currently supported in Windows ME only.

## USB Keyboard Report

The keyboard implements two HID interfaces on endpoint 1 and 2 in a USB composite-device fashion. HID interface 0 (endpoint 1) implements a standard HID keyboard with identical report and boot protocols. HID interface 1 (endpoint 2) implements multimedia and power management keys. This implementation ensures the keyboard works in BIOS setup and in DOS mode.

Interface 0 will issue 8-byte input reports that are identical to the standard keyboard boot protocol report (see Table 1-9) as documented in the Device Class Definition for Human Interface Device (HID) version 1.1. This interface also allows the host system to turn on and off the respective LED state indicators, as specified by the 1-byte output report (see Table 1-10).

**Table 1-9. Interface 0 Input Report**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| **0** | Right GUI | Right ALT | Right Shift | Right Control | Left GUI | Left ALT | Left Shift | Left Control |
| **1** | Reserved | | | | | | | |
| **2** | Keyboard Usage ID (Key Code) | | | | | | | |
| **3** | Keyboard Usage ID (Key Code) | | | | | | | |
| **4** | Keyboard Usage ID (Key Code) | | | | | | | |
| **5** | Keyboard Usage ID (Key Code) | | | | | | | |
| **6** | Keyboard Usage ID (Key Code) | | | | | | | |
| **7** | Keyboard Usage ID (Key Code) | | | | | | | |

**Table 1-10. Interface 0 Output Report**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | | | | Kana | Compose | Scroll Lock | Caps Lock | Num Lock |

Interface 1 issues power management key or multimedia key input reports, which are distinguished by a unique Report ID. The power management key uses Report ID number 1 and the multimedia key uses Report ID number 2 (see Table 1-11 and Table 1-12).

**Table 1-11. Interface 1 Power Key Input Report**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Report ID = 1 | | | | | | | |
| 1 | | | | | | Power | Wake | Sleep |

**Table 1-12. Interface 1 Multimedia Key Input Report**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Report ID = 2 | | | | | | | |
| 1 | M7 | M6 | M5 | M4 | M3 | M2 | M1 | M0 |
| 2 | M15 | M14 | M13 | M12 | M11 | M10 | M9 | M8 |
| 3 | M23 | M22 | M21 | M20 | M19 | M18 | M17 | M16 |
| 4 | Reserved for M24–M31 | | | | | | | |

Table 1-13 shows some input report examples. Report ID is not used in interface 0. The first byte is the modifier byte and is set on bit basis. Whenever a modifier key is pressed, the corresponding bit is set to one. For example, if the Left Control and the character 'A' keys are pressed, the first byte of the report equals $01, the second byte is reserved, the third byte equals $04, and the forth to the eight bytes equal $00.

Power Management keys are reported through interface 1 with report ID 1. For example, if the Wake key is pressed, the first byte of the report ID equals $01, and the second byte equals $02 since the Wake key is defined as bit 2 of the second byte

Hot keys are reported through interface 1 with reported ID 2. For example, if hot key 0 and hot key 17 are pressed, the first byte of report ID equals $02, the second byte equals $01 since hot key 0 is pressed, the third byte equals $00 since hot keys 8 to 16 are not pressed, and the forth byte equals $02 since the hot key 17 is pressed.

**Table 1-13. Input Report Examples**

| Keys Pressed | Endpoint | In Report Data |
|---|---|---|
| Left Control, 'A' | 1 | $01,$00,$04,$00,$00,$00,$00,$00 |
| Left Control, Right Alt, 'A', 'B' | 1 | $41,$00,$04,$05,$00,$00,$00,$00 |
| Wake | 2 | $01,$02 |
| Hot Key 0 & Hot Key 17 | 2 | $02,$01,$00,$02 |

## 1.6 Firmware Files

Firmware is compiled under CASM08Z.EXE ver 3.16 from P&E Microcomputer Systems, Inc.

Table 1-14 summarizes the functions of each firmware files:

**Table 1-14. Input Report Examples**

| Files | Functions |
|---|---|
| JB8-PSU.ASM | Define constants and variables<br>USB and PS/2 detection<br>PS/2 main program<br>PS/2 key handler |
| PS2-SCAN.ASM | PS/2 key scan |
| PS2-KEY.ASM | PS/2 key matrix definition |
| JB8-USB.ASM | USB main program<br>USB endpoint 1 and endpoint 2 transmit setting<br>Suspend and Resume Handler<br>Timer interrupt |
| USB-SCAN.ASM | USB key scan<br>USB key handler |
| USB-KEY.ASM | USB key matrix |
| HID-KBD.ASM | USB standard device requests handler<br>HID class requests handler |
| JB8-INT.ASM | USB interrupt<br>USB control transfer handler |
| JB8-KBD.H | Device, configure, interface, HID, endpoint, string and report descriptors |
| JB8-EQS.H | JB8 registers and memory definitions |

## 1.7 Test Description

- The solution was tested under different Windows operating systems on several brands of PCs.
- USB compliance test using Command Verifier beta version.
- Compatibility tests under Windows 98SE, 2000 and XP.
- Compatibility tests under AMD 750, Intel 810 and 845 chip set Desktops, and IBM Thinkpad 570, 600E, 600X and T23.

## 1.8 Customization

### 1.8.1 Hardware

- Leave unused row and column lines unconnected since they are pulled high by internal resistors.

### 1.8.2 Firmware

- Modify the key matrix tables in "ps2-key.asm" and "usb-key.asm" according to customized key matrix layout
- Change vendor ID, product ID and product revision number in the device descriptor table in "usb-key.h"
- Change vendor name and product name in the string descriptor table in "usb-key.h"
- Change the report descriptor in "usb-key.h" if necessary.

## 1.9 Extra Features

## 1.10 Further Information

### 1.10.1 Related Documents

MC68HC908JB8 Technical Data

Device Class Definition for Human Interface Device (HID), Version 1.1

Keyboard Scan Code Specification from Microsoft

Support for Enhanced Keyboard Features under Windows 2000 and Windows ME

USB HID Usage Table

## 1.11 Schematics

Figure 1-16 shows the schematics for the reference design.

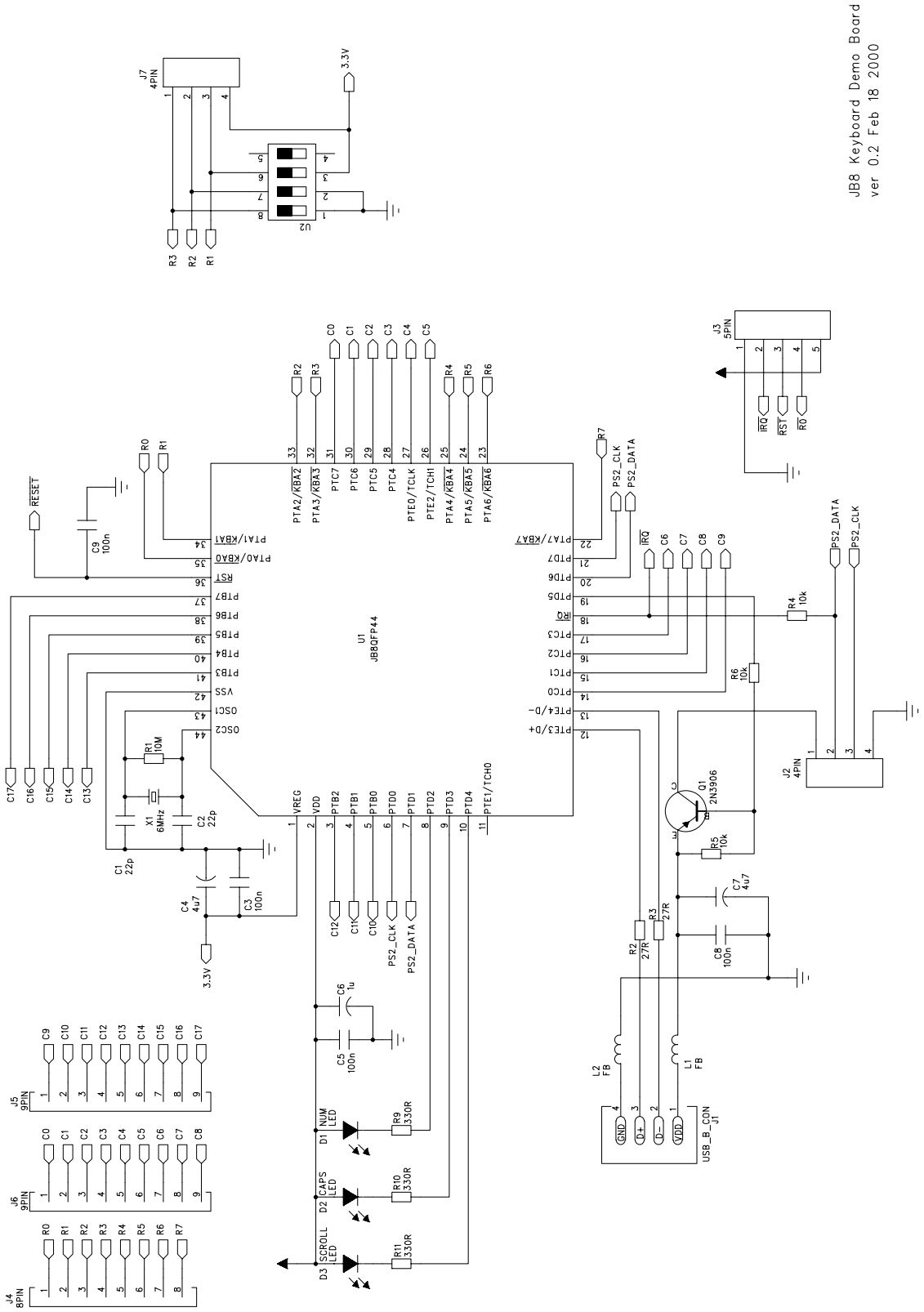


**Figure 1-16. Keyboard Schematics**

# Appendix A.
# Glossary

**A** — See “accumulator (A).”

**accumulator (A)** — An 8-bit general-purpose register in the CPU08. The CPU08 uses the accumulator to hold operands and results of arithmetic and logic operations.

**acquisition mode** — A mode of PLL operation during startup before the PLL locks on a frequency. Also see "tracking mode."

**address bus** — The set of wires that the CPU or DMA uses to read and write memory locations.

**addressing mode** — The way that the CPU determines the operand address for an instruction. The M68HC08 CPU has 16 addressing modes.

**ALU** — See “arithmetic logic unit (ALU).”

**arithmetic logic unit (ALU)** — The portion of the CPU that contains the logic circuitry to perform arithmetic, logic, and manipulation operations on operands.

**asynchronous** — Refers to logic circuits and operations that are not synchronized by a common reference signal.

**baud rate** — The total number of bits transmitted per unit of time.

**BCD** — See “binary-coded decimal (BCD).”

**binary** — Relating to the base 2 number system.

**binary number system** — The base 2 number system, having two digits, 0 and 1. Binary arithmetic is convenient in digital circuit design because digital circuits have two permissible voltage levels, low and high. The binary digits 0 and 1 can be interpreted to correspond to the two digital voltage levels.

**binary-coded decimal (BCD)** — A notation that uses 4-bit binary numbers to represent the 10 decimal digits and that retains the same positional structure of a decimal number. For example,

234 (decimal) = 0010 0011 0100 (BCD)

**bit** — A binary digit. A bit has a value of either logic 0 or logic 1.

**branch instruction** — An instruction that causes the CPU to continue processing at a memory location other than the next sequential address.

**break module** — A module in the M68HC08 Family. The break module allows software to halt program execution at a programmable point in order to enter a background routine.

**breakpoint** — A number written into the break address registers of the break module. When a number appears on the internal address bus that is the same as the number in the break address registers, the CPU executes the software interrupt instruction (SWI).

**break interrupt** — A software interrupt caused by the appearance on the internal address bus of the same value that is written in the break address registers.

**bus** — A set of wires that transfers logic signals.

**bus clock** — The bus clock is derived from the CGMOUT output from the CGM. The bus clock frequency, $f_{op}$, is equal to the frequency of the oscillator output, CGMXCLK, divided by four.

**byte** — A set of eight bits.

**C** — The carry/borrow bit in the condition code register. The CPU08 sets the carry/borrow bit when an addition operation produces a carry out of bit 7 of the accumulator or when a subtraction operation requires a borrow. Some logical operations and data manipulation instructions also clear or set the carry/borrow bit (as in bit test and branch instructions and shifts and rotates).

**CCR** — See “condition code register.”

**central processor unit (CPU)** — The primary functioning unit of any computer system. The CPU controls the execution of instructions.

**CGM** — See “clock generator module (CGM).”

**clear** — To change a bit from logic 1 to logic 0; the opposite of set.

**clock** — A square wave signal used to synchronize events in a computer.

**clock generator module (CGM)** — A module in the M68HC08 Family. The CGM generates a base clock signal from which the system clocks are derived. The CGM may include a crystal oscillator circuit and or phase-locked loop (PLL) circuit.

**comparator** — A device that compares the magnitude of two inputs. A digital comparator defines the equality or relative differences between two binary numbers.

**computer operating properly module (COP)** — A counter module in the M68HC08 Family that resets the MCU if allowed to overflow.

**condition code register (CCR)** — An 8-bit register in the CPU08 that contains the interrupt mask bit and five bits that indicate the results of the instruction just executed.

**control bit** — One bit of a register manipulated by software to control the operation of the module.

**control unit** — One of two major units of the CPU. The control unit contains logic functions that synchronize the machine and direct various operations. The control unit decodes instructions and generates the internal control signals that perform the requested operations. The outputs of the control unit drive the execution unit, which contains the arithmetic logic unit (ALU), CPU registers, and bus interface.

**COP** — See "computer operating properly module (COP)."

**counter clock** — The input clock to the TIM counter. This clock is the output of the TIM prescaler.

**CPU** — See “central processor unit (CPU).”

**CPU08** — The central processor unit of the M68HC08 Family.

**CPU clock** — The CPU clock is derived from the CGMOUT output from the CGM. The CPU clock frequency is equal to the frequency of the oscillator output, CGMXCLK, divided by four.

**CPU cycles** — A CPU cycle is one period of the internal bus clock, normally derived by dividing a crystal oscillator source by two or more so the high and low times will be equal. The length of time required to execute an instruction is measured in CPU clock cycles.

**CPU registers** — Memory locations that are wired directly into the CPU logic instead of being part of the addressable memory map. The CPU always has direct access to the information in these registers. The CPU registers in an M68HC08 are:

A (8-bit accumulator)

H:X (16-bit index register)

SP (16-bit stack pointer)

PC (16-bit program counter)

CCR (condition code register containing the V, H, I, N, Z, and C bits)

**CSIC** — customer-specified integrated circuit

**cycle time** — The period of the operating frequency: $t_{CYC} = 1/f_{OP}$.

**decimal number system** — Base 10 numbering system that uses the digits zero through nine.

**direct memory access module (DMA)** — A M68HC08 Family module that can perform data transfers between any two CPU-addressable locations without CPU intervention. For transmitting or receiving blocks of data to or from peripherals, DMA transfers are faster and more code-efficient than CPU interrupts.

**DMA** — See "direct memory access module (DMA)."

**DMA service request** — A signal from a peripheral to the DMA module that enables the DMA module to transfer data.

**duty cycle** — A ratio of the amount of time the signal is on versus the time it is off. Duty cycle is usually represented by a percentage.

**EEPROM** — Electrically erasable, programmable, read-only memory. A nonvolatile type of memory that can be electrically reprogrammed.

**EPROM** — Erasable, programmable, read-only memory. A nonvolatile type of memory that can be erased by exposure to an ultraviolet light source and then reprogrammed.

**exception** — An event such as an interrupt or a reset that stops the sequential execution of the instructions in the main program.

**external interrupt module (IRQ)** — A module in the M68HC08 Family with both dedicated external interrupt pins and port pins that can be enabled as interrupt pins.

**fetch** — To copy data from a memory location into the accumulator.

**firmware** — Instructions and data programmed into nonvolatile memory.

**free-running counter** — A device that counts from zero to a predetermined number, then rolls over to zero and begins counting again.

**full-duplex transmission** — Communication on a channel in which data can be sent and received simultaneously.

**H** — The upper byte of the 16-bit index register (H:X) in the CPU08.

**H** — The half-carry bit in the condition code register of the CPU08. This bit indicates a carry from the low-order four bits of the accumulator value to the high-order four bits. The half-carry bit is required for

binary-coded decimal arithmetic operations. The decimal adjust accumulator (DAA) instruction uses the state of the H and C bits to determine the appropriate correction factor.

**hexadecimal** — Base 16 numbering system that uses the digits 0 through 9 and the letters A through F.

**high byte** — The most significant eight bits of a word.

**illegal address** — An address not within the memory map

**illegal opcode** — A nonexistent opcode.

**I** — The interrupt mask bit in the condition code register of the CPU08. When I is set, all interrupts are disabled.

**index register (H:X)** — A 16-bit register in the CPU08. The upper byte of H:X is called H. The lower byte is called X. In the indexed addressing modes, the CPU uses the contents of H:X to determine the effective address of the operand. H:X can also serve as a temporary data storage location.

**input/output (I/O)** — Input/output interfaces between a computer system and the external world. A CPU reads an input to sense the level of an external signal and writes to an output to change the level on an external signal.

**instructions** — Operations that a CPU can perform. Instructions are expressed by programmers as assembly language mnemonics. A CPU interprets an opcode and its associated operand(s) and instruction.

**interrupt** — A temporary break in the sequential execution of a program to respond to signals from peripheral devices by executing a subroutine.

**interrupt request** — A signal from a peripheral to the CPU intended to cause the CPU to execute a subroutine.

**I/O** — See “input/output (I/0).”

**IRQ** — See "external interrupt module (IRQ)."

**jitter** — Short-term signal instability.

**latch** — A circuit that retains the voltage level (logic 1 or logic 0) written to it for as long as power is applied to the circuit.

**latency** — The time lag between instruction completion and data movement.

**least significant bit (LSB)** — The rightmost digit of a binary number.

**logic 1** — A voltage level approximately equal to the input power voltage ($V_{DD}$).

**logic 0** — A voltage level approximately equal to the ground voltage ($V_{SS}$).

**low byte** — The least significant eight bits of a word.

**low voltage inhibit module (LVI)** — A module in the M68HC08 Family that monitors power supply voltage.

**LVI** — See "low voltage inhibit module (LVI)."

**M68HC08** — A Motorola family of 8-bit MCUs.

**mark/space** — The logic 1/logic 0 convention used in formatting data in serial communication.

**mask** — 1. A logic circuit that forces a bit or group of bits to a desired state. 2. A photomask used in integrated circuit fabrication to transfer an image onto silicon.

**mask option** — A optional microcontroller feature that the customer chooses to enable or disable.

**mask option register (MOR)** — An EPROM location containing bits that enable or disable certain MCU features.

**MCU** — Microcontroller unit. See “microcontroller.”

**memory location** — Each M68HC08 memory location holds one byte of data and has a unique address. To store information in a memory location, the CPU places the address of the location on the address bus, the data information on the data bus, and asserts the write signal. To read information from a memory location, the CPU places the address of the location on the address bus and asserts the read signal. In response to the read signal, the selected memory location places its data onto the data bus.

**memory map** — A pictorial representation of all memory locations in a computer system.

**microcontroller** — Microcontroller unit (MCU). A complete computer system, including a CPU, memory, a clock oscillator, and input/output (I/O) on a single integrated circuit.

**modulo counter** — A counter that can be programmed to count to any number from zero to its maximum possible modulus.

**monitor ROM** — A section of ROM that can execute commands from a host computer for testing purposes.

**MOR** — See "mask option register (MOR)."

**most significant bit (MSB)** — The leftmost digit of a binary number.

**multiplexer** — A device that can select one of a number of inputs and pass the logic level of that input on to the output.

**N** — The negative bit in the condition code register of the CPU08. The CPU sets the negative bit when an arithmetic operation, logical operation, or data manipulation produces a negative result.

**nibble** — A set of four bits (half of a byte).

**object code** — The output from an assembler or compiler that is itself executable machine code, or is suitable for processing to produce executable machine code.

**opcode** — A binary code that instructs the CPU to perform an operation.

**open-drain** — An output that has no pullup transistor. An external pullup device can be connected to the power supply to provide the logic 1 output voltage.

**operand** — Data on which an operation is performed. Usually a statement consists of an operator and an operand. For example, the operator may be an add instruction, and the operand may be the quantity to be added.

**oscillator** — A circuit that produces a constant frequency square wave that is used by the computer as a timing and sequencing reference.

**OTPROM** — One-time programmable read-only memory. A nonvolatile type of memory that cannot be reprogrammed.

**overflow** — A quantity that is too large to be contained in one byte or one word.

**page zero** — The first 256 bytes of memory (addresses $0000–$00FF).

**parity** — An error-checking scheme that counts the number of logic 1s in each byte transmitted. In a system that uses odd parity, every byte is expected to have an odd number of logic 1s. In an even parity system, every byte should have an even number of logic 1s. In the transmitter, a parity generator appends an extra bit to each byte to make the number of logic 1s odd for odd parity or even for even parity. A parity checker in the receiver counts the number of logic 1s in each byte. The parity checker generates an error signal if it finds a byte with an incorrect number of logic 1s.

**PC** — See “program counter (PC).”

**peripheral** — A circuit not under direct CPU control.

**phase-locked loop (PLL)** — A oscillator circuit in which the frequency of the oscillator is synchronized to a reference signal.

**PLL** — See "phase-locked loop (PLL)."

**pointer** — Pointer register. An index register is sometimes called a pointer register because its contents are used in the calculation of the address of an operand, and therefore points to the operand.

**polarity** — The two opposite logic levels, logic 1 and logic 0, which correspond to two different voltage levels, $V_{DD}$ and $V_{SS}$.

**polling** — Periodically reading a status bit to monitor the condition of a peripheral device.

**port** — A set of wires for communicating with off-chip devices.

**prescaler** — A circuit that generates an output signal related to the input signal by a fractional scale factor such as 1/2, 1/8, 1/10 etc.

**program** — A set of computer instructions that cause a computer to perform a desired operation or operations.

**program counter (PC)** — A 16-bit register in the CPU08. The PC register holds the address of the next instruction or operand that the CPU will use.

**pull** — An instruction that copies into the accumulator the contents of a stack RAM location. The stack RAM address is in the stack pointer.

**pullup** — A transistor in the output of a logic gate that connects the output to the logic 1 voltage of the power supply.

**pulse-width** — The amount of time a signal is on as opposed to being in its off state.

**pulse-width modulation (PWM)** — Controlled variation (modulation) of the pulse width of a signal with a constant frequency.

**push** — An instruction that copies the contents of the accumulator to the stack RAM. The stack RAM address is in the stack pointer.

**PWM period** — The time required for one complete cycle of a PWM waveform.

**RAM** — Random access memory. All RAM locations can be read or written by the CPU. The contents of a RAM memory location remain valid until the CPU writes a different value or until power is turned off.

**RC circuit** — A circuit consisting of capacitors and resistors having a defined time constant.

**read** — To copy the contents of a memory location to the accumulator.

**register** — A circuit that stores a group of bits.

**reserved memory location** — A memory location that is used only in special factory test modes. Writing to a reserved location has no effect. Reading a reserved location returns an unpredictable value.

**reset** — To force a device to a known condition.

**ROM** — Read-only memory. A type of memory that can be read but cannot be changed (written). The contents of ROM must be specified before manufacturing the MCU.

**SCI** — See "serial communication interface module (SCI)."

**serial** — Pertaining to sequential transmission over a single line.

**serial communications interface module (SCI)** — A module in the M68HC08 Family that supports asynchronous communication.

**serial peripheral interface module (SPI)** — A module in the M68HC08 Family that supports synchronous communication.

**set** — To change a bit from logic 0 to logic 1; opposite of clear.

**shift register** — A chain of circuits that can retain the logic levels (logic 1 or logic 0) written to them and that can shift the logic levels to the right or left through adjacent circuits in the chain.

**signed** — A binary number notation that accommodates both positive and negative numbers. The most significant bit is used to indicate whether the number is positive or negative, normally logic 0 for positive and logic 1 for negative. The other seven bits indicate the magnitude of the number.

**software** — Instructions and data that control the operation of a microcontroller.

**software interrupt (SWI)** — An instruction that causes an interrupt and its associated vector fetch.

**SPI** — See "serial peripheral interface module (SPI)."

**stack** — A portion of RAM reserved for storage of CPU register contents and subroutine return addresses.

**stack pointer (SP)** — A 16-bit register in the CPU08 containing the address of the next available storage location on the stack.

**start bit** — A bit that signals the beginning of an asynchronous serial transmission.

**status bit** — A register bit that indicates the condition of a device.

**stop bit** — A bit that signals the end of an asynchronous serial transmission.

**subroutine** — A sequence of instructions to be used more than once in the course of a program. The last instruction in a subroutine is a return from subroutine (RTS) instruction. At each place in the main program where the subroutine instructions are needed, a jump or branch to subroutine (JSR or BSR) instruction is used to call the subroutine. The CPU leaves the flow of the main program to execute the instructions in the subroutine. When the RTS instruction is executed, the CPU returns to the main program where it left off.

**synchronous** — Refers to logic circuits and operations that are synchronized by a common reference signal.

**TIM** — See "timer interface module (TIM)."

**timer interface module (TIM)** — A module used to relate events in a system to a point in time.

**timer** — A module used to relate events in a system to a point in time.

**toggle** — To change the state of an output from a logic 0 to a logic 1 or from a logic 1 to a logic 0.

**tracking mode** — Mode of low-jitter PLL operation during which the PLL is locked on a frequency. Also see "acquisition mode."

**two's complement** — A means of performing binary subtraction using addition techniques. The most significant bit of a two's complement number indicates the sign of the number (1 indicates negative). The two's complement negative of a number is obtained by inverting each bit in the number and then adding 1 to the result.

**unbuffered** — Utilizes only one register for data; new data overwrites current data.

**unimplemented memory location** — A memory location that is not used. Writing to an unimplemented location has no effect. Reading an unimplemented location returns an unpredictable value. Executing an opcode at an unimplemented location causes an illegal address reset.

**V** —The overflow bit in the condition code register of the CPU08. The CPU08 sets the V bit when a two's complement overflow occurs. The signed branch instructions BGT, BGE, BLE, and BLT use the overflow bit.

**variable** — A value that changes during the course of program execution.

**VCO** — See "voltage-controlled oscillator."

**vector** — A memory location that contains the address of the beginning of a subroutine written to service an interrupt or reset.

**voltage-controlled oscillator (VCO)** — A circuit that produces an oscillating output signal of a frequency that is controlled by a dc voltage applied to a control input.

**waveform** — A graphical representation in which the amplitude of a wave is plotted against time.

**wired-OR** — Connection of circuit outputs so that if any output is high, the connection point is high.

**word** — A set of two bytes (16 bits).

**write** — The transfer of a byte of data from the CPU to a memory location.

**X** — The lower byte of the index register (H:X) in the CPU08.

**Z** — The zero bit in the condition code register of the CPU08. The CPU08 sets the zero bit when an arithmetic operation, logical operation, or data manipulation produces a result of $00.

## *How to Reach Us:*

**Home Page:**
www.freescale.com

**E-mail:**
support@freescale.com

**USA/Europe or Locations Not Listed:**
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

**Europe, Middle East, and Africa:**
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

**Japan:**
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

**Asia/Pacific:**
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

***For Literature Requests Only:***
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics of their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to http://www.freescale.com/epp.

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2003, 2004, 2006. All rights reserved.

DRM014
Rev. 1.0, 4/2006

freescale™
semiconductor